

Recursion, Exception Handling in C++, and Linked List

Objectives:

- Practice using Recursion to traverse Linked Lists.
- Practice refactoring operations (refactoring iterative solutions to recursive implementations.)
- Working with projects with multiple files.
- Program comprehensions, debugging, and refactoring
- Practice using exceptions and exception handling in C++

Note: This example is very similar or at least useful for this part:

[CS244 002 Spr2021 RecursiveWithNotelistAndIteratorsExamples.zip](#)

In this project, you are given a partial implementation of an accounting system (a subsystem of a full system) that controls the customer's balances within a hypothetical bank. The system is expected to help the bank automate some of its common query processes (without getting involved in the customer's detailed information). It uses a data structure to keep track of the bank accounts in the system. Moreover, basic account information (names and current balances) is already provided in a text file, and no need to worry about inserting new accounts into the file. However, the system needs to conduct some queries and report generations. The good news is that most of the work is done on your behalf (in iterative ways).

The given program performs at least the following operations:

1. Print All accounts. (available and already reserved)
2. Print names of the customer who have balances below a given amount.
3. The customer with the highest balance. (Max)
4. Exit

For simplicity, only two major components (classes) were used to develop the system, **BankAccount**, and **Bank** with a list of customer accounts stored in a text file.

An application file that contains the main function is provided as well. The main function controls the selection menu and passing the information file to the right object (bank).

You need to understand (comprehend) how the system works in order to be able to complete the refactoring task.

Data structures used in the system: Doubly-Linked List ADT.

Your job:

Task 1: Refactoring: reimplementing (rewriting) the following methods recursively (using **recursion**).

- *`void Bank::printAllAccounts();`*
- *`void Bank::printAccountsBelowAmount(double amount);`*
- *`BankAccount Bank::getMaxAccount();`*

Notice that refactoring a method to be implemented recursively can sometimes involve changing the parameter list. However, the function name and the returned value preferred to be the same. For example, the signature of the recursive version of the method getMaxAccount() could become like the following:

BankAccount getMaxAccount(DNode<BankAccount>*);

Hint: The caller of the method above needs to send the head of the doubly-linked list that is a member of the class Bank.

Task 2: In this part, you are expected to enhance (improve) the system with exception handling. That is, you need to handle some of the operations that can cause a problem during the runtime using exceptions.

Below are some of the operations you need to improve by using exceptions:

Entering an invalid integer entry (other than 0 to 4)

File not found or cannot be open for reading.

You need to show what other operations can be improved with exception handling. Think about a situation where a non-integer value is entered by the user.

Supporting Code Files

- Complete header files that you need to study before working on the implementation. The file contains two classes and relevant friend functions.
- The full implementation of a simple Doubly-Linked List ADT is posted on Canvas. It was used to create underlining data structures in the second class (Bank) for the accounts list.
- Full implementation for the main function with the operation menu. Will need to be updated as the methods will probably change when you rewrite them in a recursive way.
- A text file with sample records
- Please go over the slides on recursion posted on Canvas to see how linked lists are traversed recursively.
- Read the materials and watch the recordings on Exception handling if needed.

- **A zipped incomplete project is provided on Canvas.** [assignment4 cs244 fall2021.zip](#)
[Download assignment4 cs244 fall2021.zip](#)




Please never hesitate to contact your instructors (Dr. Alnaeli and Mr. Zach Boehm) should you have any questions. Use our office hours and tutoring service.

Dr. Alnaeli

Ziped Project that needs to be unzipped and then undergoes refactoring:

[assignment4_cs244_fall2021.zip](#)  [Download assignment4_cs244_fall2021.zip](#)

Source Code (individual files if that works better for you): [assignment4_cs244_fall2021.zip](#) 
[Download assignment4_cs244_fall2021.zip](#)

[applicationMainFile.cpp](#)  [Download applicationMainFile.cpp](#)

[bank_H.h](#)  [Download bank_H.h](#)

[doublyLinkedList_H.h](#)  [Download doublyLinkedList_H.h](#)

[database_names_balances.txt](#)  [Download database_names_balances.txt](#)